Hierarchical Modelling of HW/SW Control Systems: an Application to Dependability Analysis of Railway Interlocking

M. Nelli, A. Bondavalli e L. Simoncini

CNUCE: C96 - 09

Hierarchical Modelling of HW/SW Control Systems: an Application to Dependability Analysis of Railway Interlocking

M. Nelli*, A. Bondavalli, L. Simoncini***

* Dip. Ingegneria Informazione, Univ. Pisa, Pisa, Italy

** CNUCE - CNR, via S. Maria 36, 56126, Pisa, Italy,

Abstract

This paper describes the dependability modelling and evaluation of a real complex system, made of redundant replicated hardware and redundant diverse software. It takes into account all aspects of their interactions (including correlation between the diverse software variants) and of the criticality of the several components. Our approach has been to realise the system model in a structured way. This allows to cope with complexity and to focus, where interesting, on specific behaviour for a more detailed analysis. Furthermore each level may be modelled using different methodologies and its evaluation performed with different tools without the need of modifying the general structure of the model. In order to validate the most complex sub-models, we built alternatives using different tools and methodologies; this proved to be very useful since it allowed to find small bugs and imperfections and to gain more confidence that the models represented the real system behaviour.
With respect to the real system taken as the example, our analyses allowed to establish the dependability bottlenecks of the current version and to state targets for the several sub-components such that the system targets could be reached, thus providing hints for next releases or modifications of the system and information to assign targets to the various components of the system.

Key-words: Complex Control Systems, Dependability Evaluation, Markov Models, Stochastic Activity Networks.

1. Introduction

Railway station interlocking systems based on microprocessors were developed in all technologically advanced countries and have been used since a few years by those Railway Authorities wishing to have a good cost/benefit ratio. In Europe and in Japan solid state interlocking systems were used in passenger transportation networks with medium/large stations and heavy-medium range traffic; in these applications complex interlocking systems were designed, including central and remote peripheral units, with vital data transmission between them [15], [7], [17], [18], [19], [22], while in the US small systems have been produced since the 80's, usually applied to freight transportation lines [3], [10] .

The use of computer controlled interlocking systems, in the place of the usual electro-mechanical systems, introduces non trivial problems in their design and analysis. Most difficult are those parts of the systems delegated to the control of vital functions, where the interactions between the redundant hardware and the application software have a critical impact on system safety. These interactions have an impact on modelling complexity since they induce stochastic dependencies that must be taken into account in modelling the behaviour of components and their interactions. In the literature several papers exist in the field of dependability analysis [1], [20], [5], [4], and some basilar papers exist on the approach to dependability evaluation of combined hardware and software systems [6], [9], [12], [13], [8], but detailed modelling of the interactions between hardware and software components, in particular for critical systems, and the influence of the related dependencies has been treated, at our knowledge, only in [11].

The interest of such modelling lies in the support it may provide in the design phase of a complex system, when decisions have to be taken on possible different structures of the system for matching the dependability requirements imposed by the regulatory authorities. In the design phase it may be cost beneficial to construct different models for the different architectures and the several alternatives can be quantitatively evaluated; in this way sensitivity analysis is possible, to ascertain what are the most important parts of a design on which more resources have to be spent than on others and to identify, in a statistical manner, the dependability levels of the several hardware or software components and the trade-offs between them. This type of analysis, made for an already existing system, as it is the case in this paper, is important for an "a posteriori" dependability evaluation, for pointing out possible design weak points or bottlenecks, for the late validation of the dependability requirements (this can also be useful in certifying phase) and to provide sound hints for next releases or modifications of the systems.

The contribution of this paper is the modelling of a real complex systems, made of redundant replicated hardware and redundant diverse software taking into account all aspects of their interactions (including correlation between the diverse software variants) and of the criticality of the several components. Our approach has been to realise the system model in a structured way. This allows to cope with complexity and to focus, where interesting, on specific behaviour for a more detailed analysis. Structuring in different levels separated by well identified interfaces allows to realise each level with different methodologies and to perform its evaluation with different tools without the need of modifying the general structure of the model. Each level has been subdivided into several sub-levels for a finer analysis of some characteristics. The higher level of the hierarchy is made of the models for the evaluation of the dependability measures of inter-

est (in our case, beside the availability, also reliability and safety measures have been assessed). These models use the values of success or failure probability obtained by the modelling of a mission, which describes the system behaviour on a period of time. The model of a mission uses, in its turn, the values obtained by several different models of a single execution and finally the model of one execution is subdivided into other levels which take into account the specific behaviour of the components of the system. With this structuring, each level is a sort of abstract object, whose implementation details are transparent to the adjacent levels, and therefore can be realised and analysed using the most proper tools and methodologies, which can differ from one level to another. Despite our effort for reducing the complexity of the individual levels with respect to the complexity of the entire system, some complex level remained and has been realised using different methodologies to compare and validate the used model.

The paper is organised as follows. Section 2 contains an overview of the Ansaldo TMR MODIAC system; Section 3 defines the meaning of the basic parameters used and describes our assumptions and the modelling approach. Sections 4 and 5 contain a description of the various models for one execution and for the mission respectively. A few evaluations of the dependability attributes are shown in Section 6 and finally Section 7 concludes this paper.

2. Ansaldo TMR MODIAC system.

We analysed the "Apparato Centrale con Calcolatore" (ACC) [15] system nucleus constructed by Ansaldo for their railway station signalling control system, called TMR MODIAC. The system is divided into two parts (Figure 1):

- **SN** -->> is the subsystem which performs vital functions: it comprises the Safety Nucleus and a variable number, depending on the station size, of distributed Trackside Units, which communicate the state of the station to the central computer. After the necessary interactions with the operator, processing and controls, the central computer sends back commands for the signalling system to the Trackside Units.
- **RDT** -->> is a supervisory subsystem that performs Recording, Diagnosis and Telecontrol functions; this subsystem allows continuos monitoring of the system state and events recording; the latter is useful to make estimations and find out less reliable sections.

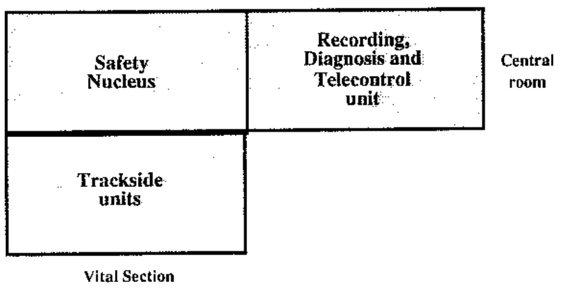

Figure 1: The TMR MODIAC system

The Safety Nucleus [14] is structured as shown in Figure 2 and is the vital part of the system. It comprises six units with a separated power supply unit. The three Nsi sections represent three computers which are connected in TMR configuration, i.e. working with a "2-out-of-3" major-

ity: three diverse software programs performing iteratively the same tasks, run inside three identical hardware sections. The system is designed to keep on running even after the failure of one section; in such a case the section is excluded and the system uses only two sections with a "2-out-of-2" majority, until the failed section is restored. A section excluded after failing is restored after a few minutes. The Exclusion Logic is a fail-safe circuit whose job is to electrically isolate the section that TMR indicated to be excluded. The activation/de-activation unit is a device that switches on and controls power supply units. The video switching unit controls video images to be transmitted to the monitors of the operator terminal.

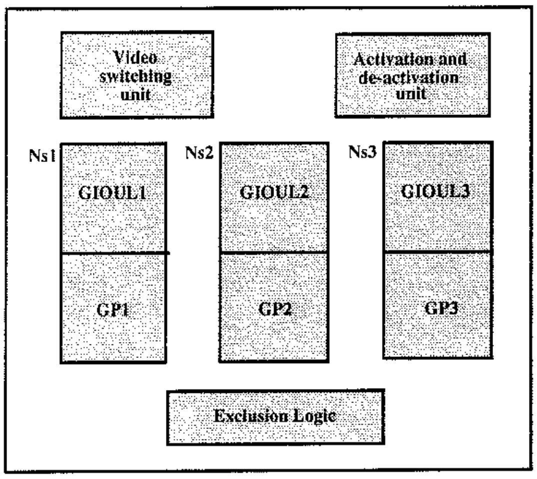

Figure 2: Structure of the Safety Nucleus

The TMR sections carry out the same functions; the hardware, the basic software architecture and the operating environment are exactly the same; while "design diversity" [2] was adopted in the development of software application modules. Each section is composed by two physically separated units which carry out different functions in parallel:

- **GIOUL** (operator interface manager and logical unit): executes the actual processing and manages the interactions with the Operator Terminal and the RDT subsystem;
- **GP** (trackside manager): manages the communications with the Trackside Units and modifies, whenever necessary, the commands given by GIOUL.

The processing loops last 1 second for GIOUL and 250 msec for GP: this causes the communications between GIOUL and GP belonging to the same TMR section to be performed at every second (GIOUL loop), i.e. once every four loops of GP. Instead the communications between units of the same type (between the three GIOUL units and separately between the three GP units) are carried out at every processing loop.

Each TMR (GIOUL and GP separately) unit votes on the state variables and processing results. If it finds any inconsistency between its results and those of the other units and three sections are active, it can recover a presumably correct state and continue processing. If one section disagrees twice in a row it is excluded. No disagreement is tolerated when only two sections are active. Besides voting on software each unit controls communications and tests internal boards functionalities. Based on hardware test results, one section can decide to exclude itself from the system. Diagnostic tests are carried out during the usual unit operation; they are implemented

such that they do not modify the operating environment. Each section is also able to detect malfunctions on its databases thus deciding to exclude itself. In addition to these tasks, GIOUL has to manage the communications with the Operator Interface, and to perform tests on keyboard inputs as well. If an error is detected a signal is displayed.

TMR MODIAC is a critical system, meaning that failures or unavailability can have catastrophic consequences, both economical and for human life. The constraints to be satisfied by the system are a probability of catastrophic failure less than or equal to 10^{-11} per hour and no more than 5 minutes down time are allowed over 8600 hours (i.e. availability higher than or equal to 0.999990310078).

3. Assumptions and Modelling Approach.

3.1 Assumptions and Basic Parameters

We restricted our modelling effort to the Safety Nucleus, the most relevant part of the system. Our model does not include the RDT subsystem neither the Trackside Units, but represents the overall functionalities of the Nucleus, including the main features and the interactions among the different components. The main components that must be considered in modelling the system are: hardware, software, databases and, only for GIOUL, acceptance test on the input from the Operator Terminal. Hardware aspects cover internal boards and physical characteristics of the communications while software aspects cover the operating system and software modules that are sequentially activated during the processing loops. The databases, whose control represents one of the ways for detecting errors in various modules, cover both hardware and software aspects: database malfunction can be due to either corruption of memory cells or an error of the managing software. One of the tasks that GIOUL has to perform is checking the correctness of the inputs issued by the operator terminal keyboard before transmitting them to the other modules for their processing; this check is very important since it can avoid the system to send wrong commands to the Trackside Units. For this reason the software module performing this check is not considered together with the other software modules of GIOUL. We also made the choice of not modelling in detail the system while an excluded section is restored. More precisely we account for the time required for restoring a section but we neglect the particular configurations that GIOUL and GP can assume during that period.

The definition of the basic events we have considered and the symbols used to denote their probabilities are reported in Table 1. The following assumptions have been made:

1) "Compensation" among errors never happens;

2) The Video Switching, the Activation/de-Activation and the (external fail-safe) Exclusion Logic units are considered reliable;

3) The Diagnostic module, (that exploits majority voting), and Exclusion Management module within GIOUL and GP are considered reliable.

4) Two erroneous, identical outputs are only the result of correlated errors; two or three independent errors within different units are always distinguishable by the voting.

5) During one execution, both GIOUL and GP may suffer from many errors, at most one for each component (software, hardware, databases and acceptance test for GIOUL).

6) Errors affecting different components of the same unit are statistically independent.

7) The hardware communication aspects of the Nucleus are grouped together with the other hardware aspects; the software part of communications is assumed reliable.

8) The occurrence of hardware detected or undetected faults which do not prevent the correct continuation of activity is disregarded.

9) The execution of each iteration is statistically independent from the others.

10) Symmetry: the error probabilities of GIOUL and GP are the same for the three sections.

11) GIOUL units receive identical inputs from the keyboard.

Error type (Events)	**Probabilities (GIOUL)**	**Probabilities (GP)**
independent hardware fault in a unit	qhl	qhp
the diagnostic does not detect a error caused by an hardware fault	qhdl	qhdp
the diagnostic erroneously detects a (non-present) error due to independent hardware fault	qhndl	qhndp
an independent error in a database is detected	qbrl	qbrp
an independent error in a database is not detected	qbnrl	qbnrp
correlated error between three databases	q3bdl	q3bdp
correlated error between two databases	q2bdl	q2bdp
independent software error in a unit	qil	qip
correlated software error between three units	q3vl	q3vp
correlated software error between two units	q2vl	q2vp
independent error of the acceptance test in a unit (it accepts a wrong input or does not accepts a correct one)	qail	.....
correlated error between the acceptance tests of three units accepting the same wrong input	q3al	......
correlated error between the acceptance tests of two units accepting the same wrong input	q2al	......

Table 1: Basic error types and symbols used to denote their probabilities

3.2 *Modelling Approach*

The model was conceived in a modular and hierarchical fashion, structured in layers. Each layer has been structured for producing some results while hiding implementation details and internal characteristics: output values from one layer may thus be used as parameters of the next higher layer. In this way the entire modelling can be simply handled. Further, different layers can be modelled using different tools and methodologies: this leads to flexible and changeable sub-models so that one can vary the accuracy and detail with which specific aspects can be studied. The specific structure of each sub-model depends both on the system architecture and on the

measurements and evaluations to be obtained. The model of the Safety Nucleus of the TMR MODIAC we have built, shown in Figure 3, can be split into two main parts: the first part deals with one execution and computes the probabilities of success or failure; the second one, building on this, allows the evaluations of the dependability attributes for an entire mission.

In the previous section we explained that if a disagreement is found while all three sections are active, GIOUL and GP recover the correct value and participate to the next loop. This holds for one single disagreement: if one section disagrees twice in a row it is excluded at the end of the current loop. Therefore, in order to represent as close as possible the actual system behaviour, we had to make several models to keep memory of previous disagreement of one section at the beginning of the execution.

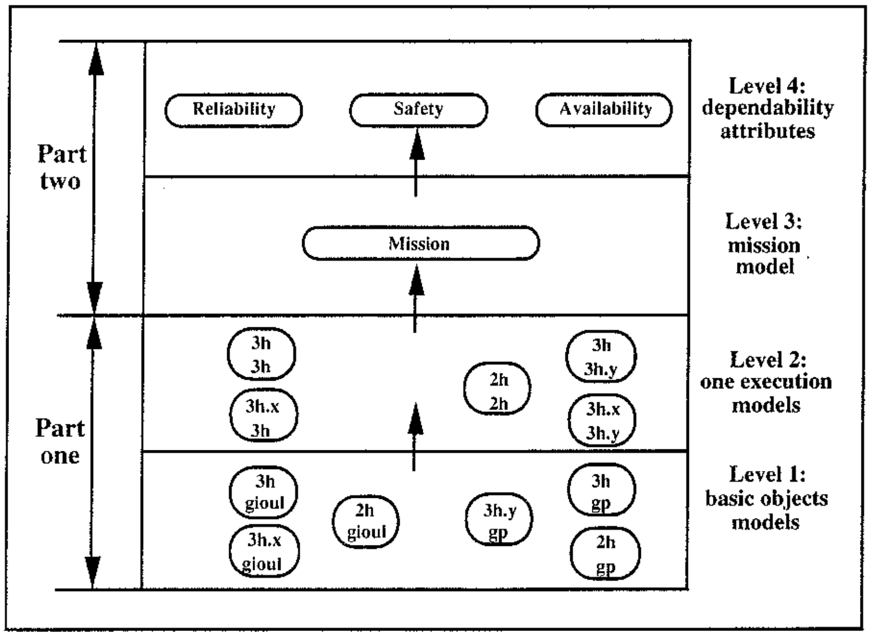

Figure 3: High level model of the TMR MODIAC

To describe the GIOUL and GP TMR units at level 1 (Figure 3) we defined:

- five sub-models of the behaviour of GP in configurations **3h**, **3h.1**, **3h.2**, **3h.3**, **2h** (**3h.x** means that section x (1, 2 or 3) disagreed during the previous loop);
- five sub-models of the behaviour of GIOUL (**3h**, **3h.1**, **3h.2**, **3h.3**, **2h**).

One system execution (level 2) lasts 1 second, it includes one GIOUL (1 second) and four GP (250 msec) iterations and could also be considered as brief one–second mission. Due to the need to keep memory of previous disagreements, 17 different models for one-execution have been defined: one models the system when only two sections are active, while the remaining describe the system with three active sections (level 2 in Figure 3):

- **3h/3h**: GIOUL and GP are correctly working at the beginning of the execution.
- **3h.x/3h**: the GIOUL of section x (1, 2 or 3) disagreed during the previous loop while GP is correctly working.

- **3h/3h.y**: GIOUL units are correctly working while the GP of section y (1, 2 or 3) disagreed during the previous (GP) execution.
- **3h.x/3h.y**: both the GIOUL of section x and the GP of section y disagreed during the previous loop (x and y can represent the same section or different ones)
- **2h/2h** : execution begins with only two active sections.

Each of the 17 models uses, in different combinations and sequences, the same base objects of level 1 and describes the essential characteristics of the Safety Nucleus. It models the functions of the units as a whole: different aspects of the units are not considered separately, instead their interactions and their peculiar nature is considered. Specifically hardware and software aspects cannot be always kept separated because on one side, hardware faults often manifest themselves as software errors and on the other, the methods for software fault tolerance allow to tolerate not only software faults but also hardware faults of the internal boards and of the communication cables. This explains why GP and GIOUL models do not comprise different separated sub-models each one regarding one single aspect, but are structured as a unique global sub-model in which both interactions and specific aspects are included. The models for one-execution are conceived to compute (and to provide to level three) the following probabilities:

- ***probability of success of one-execution***; it is the probability that the system performs an entire one second mission correctly. This implies that the system is ready to start the next execution. It is composed by many different success probabilities according to the configuration achieved.
- ***probability of safe failure of one-execution***; it is the probability that the system fails during one execution and stops avoiding catastrophic damages (this is ensured by the ACC system that is designed so that it stops when malfunctions occur, forcing devices and subsystems to lock in a safe state).
- ***probability of catastrophic failure of one-execution***; it is the probability that the Nucleus, failing, keeps on sending erroneous commands causing serious damages.

The mission model (level 3), considering all the possible system configurations during one execution, describes the system behaviour during time. Once that both the single execution and the mission models have been constructed we focused on which kind of measurements are required. For our highly critical system the following dependability attributes have been evaluated: reliability, safety and availability. While reliability and safety can be both obtained by computing the probabilities of catastrophic and safe failure at time t defined as the duration of the mission, availability required the definition of a specific availability model.

4. Models for one execution.

Two methodologies have been adopted to build the models for one-execution: Discrete time Markov chains that have been manually drawn for which the probability evaluation has been

accomplished using "Mathematica"[1], and Stochastic Activity Networks that have been directly solved using the software tool "Ultrasan"[2]. Since Markov chains are often impractical, even if they provide symbolic results, Ultrasan has been adopted in order to avoid building 17 repetitive one-execution models using only Markov chains. Only two models (3h/3h and 2h/2h) have been completely built using Markov chains in order to test and validate the results obtained by Ultrasan. This redundancy in building models has been very useful: some errors occurred during the model developing phase have been detected. Ultrasan has been a good choice, since we could develop one single model that allowed to compute the results for the 17 different scenarios. In fact, by assigning different values to the variables of the model, thus representing different initial markings, we could represent the different states of the system and account for previous failures of the various sub-components. The model is also able to distinguish the various configurations without having to replicate the unchanged aspects. Only two of the seventeen models were tested using Markov chains but those two models are the most relevant ones and cover all the scenarios that need to be represented. The results obtained by Ultrasan and Markov, using the same values for the parameters, were in agreement. Now we show, as an example, some objects belonging to the two lower levels. First the Markov chains are described and later the Ultrasan model.

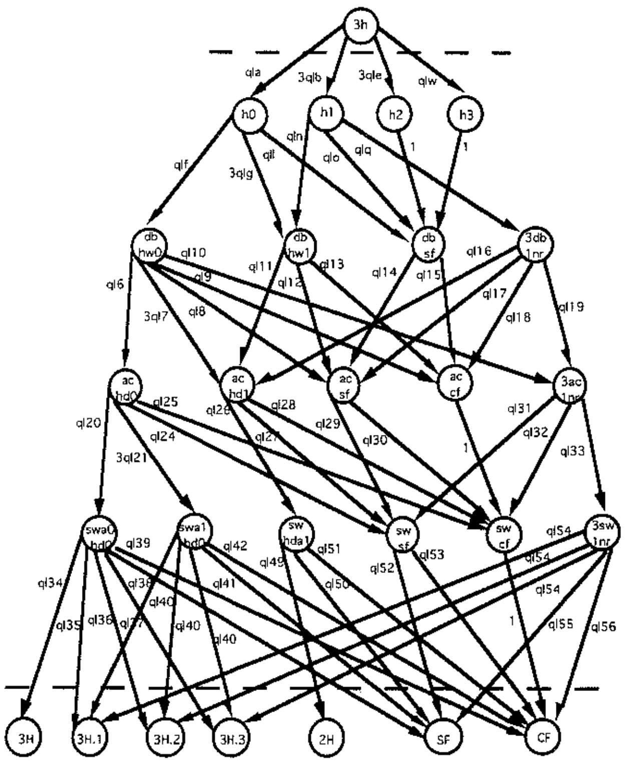

Figure 4: Detailed description of the behaviour of GIOUL in configuration 3h

[1] Mathematica, vers. 2.2, Wolfram Research, Inc.

[2] Ultrasan, ver. 3.0.1, University of Illinois at Urbana-Champaign: Center for Reliable and High-Performance Computing Coordinated Science Laboratory, 1994.

Probability notation	Expression	Probability notation	Expression
qla	(1-qhl)^3	ql16	(1-qbrl-qbnrl)^2 qbrl (1-qcbdl3)
qlb	qhl (1-qhl)^2	ql19	((1-qbrl-qbnrl)^3+ qbnrl (1-qbrl-qbnrl)^2) (1-qcbdl3)
qle	qhl^2 (1-qhl)	ql20	(1-qail)^3 (1-qacl3);
qlw	qhl^3	ql21	(1-qail)^{2}qail (1-qacl3)
qlf	(1-qhndl)^3	ql24	(3 qail^2(1-qail)+qail^3) (1-qacl3)
qlg	qhndl (1-qhndl)^2	ql25 = ql28= ql30 = ql32	qacl3
qll	3 qhndl^2 (1-qhndl)+ qhndl^3	ql26 = ql33	ql20 + ql21
qln	(1-qhdl) (1-qhndl)^2	ql27 = ql31	2 ql21 + ql24
qlo	qhdl (qhndl^2+2 qhndl (1-qhndl)) + (1-qhdl) (2 qhndl (1-qhndl) + qhndl^2)	ql29	(1-qacl3)
qlq	qhdl (1-qhndl)^2	ql34	(1-qil)^3 (1-qcl3)
ql6	(1-qbrl-qbnrl)^3 (1-qcbdl3)	ql35 = ql36 = ql37	(1-qil)^2 qil (1-qcl3)
ql7	qbrl (1-qbrl-qbnrl)^2 (1-qcbdl3)	ql38	(3 qil^2 (1-qil) + qil^3) (1-qcl3)
ql8	(3 qbrl^2(1-qbrl-qbnrl) +qbrl^3+3 qbrl^2 qbnrl +qbnrl^3+3 qbnrl^2(1-qbrl-qbnrl)+3qbnrl^2 qbrl+6 qbrl qbnrl (1-qbnrl-qbrl)) (1-qcbdl3)	ql40 = ql54	(1/3) (ql34 + ql35)
ql9 = ql13 = ql15 = ql18	qcbdl3	ql41 = ql55	ql37 + ql36 + ql38
ql10	3 qbnrl (1-qbrl-qbnrl)^2 (1-qcbdl3)	ql42 = ql56 = ql39 = ql51 = ql53	qcl3
ql11	((1-qbrl-qbnrl)^3 + qbrl (1-qbrl-qbnrl)^2 + (1-qbrl-qbnrl)^2 qbnrl) (1-qcbdl3)	ql49	ql34 + ql35
ql12	(2 qbrl(1-qbrl-qbnrl)^2 +qbrl^3+3 qbrl^2 (1-qbrl-qbnrl)+3 qbrl^2 qbnrl + qbnrl^3 + 3 qbnrl^2 qbrl+3 qbnrl^2 (1-qbrl-qbnrl)+2 (1-qbrl-qbnrl)^2 qbnrl+6 qbnrl qbrl (1-qbnrl-qbrl)) (1-qcbdl3)	ql50	ql36 + ql37 + ql38
ql14	1-qcbdl3	ql52	(1-qcl3)
ql17	(2qbrl(1-qbnrl-qbrl)^2 +qbrl^3+3qbrl^2(1-qbrl-qbnrl)+3qbrl^{2}qbnrl+qbnrl^3+3qbnrl^{2}qbrl+3 qbnrl^2(1-qbrl-qbnrl)+2(1-qbrl-qbnrl)^{2}qbnrl+6qbrl qbnrl(1-qbrl-qbnrl)) (1-qcbdl3)		

Table 2: Transition probabilities of Figure 4, according to the Table 1

State notation	Meaning
3h	start of GIOUL 3h
h0	not hardware fault
h1	hardware fault into an unit
h2	hardware fault into two units
db hw0	not hardware diagnosed fault
db hw1	hardware diagnosed fault into a unit
db sf	diagnostic tests detected failure due to hardware faults
3db 1nr	hardware not diagnosed fault that appears as a processing error
ac hd0	hardware and databases are correct
ac hd1	a unit is already failed because of a hardware or database fault
ac sf	two or three units are already failed because of hardware or database faults
ac cf	catastrophic GIOUL TMR failure due to database faults
3ac 1nr	hardware or database not diagnosed faults that appear as a processing error
swa0 hd0	hardware, databases and acceptance test are correct
swa1 hd0	hardware and databases are correct but a unit acceptance test doesn't operate correctly
sw hd1	a unit is already failed because of a hardware or database fault
sw sf	two or three units are already failed because of hardware, database or acceptance test faults
sw cf	catastrophic GIOUL TMR failure due to database or acceptance test faults
3sw 1nr	hardware or database not diagnosed fault that appears as a processing error

Table 3: Notation of the states of Figure 4

As an example of a model of the level 1 (base object level), we report in Figure 4 the detailed description representing the behaviour of the GIOUL TMR in configuration 3h (the notation of the states of Figure 4 is reported in Table 3). The models for other configurations can be obtained by analogy with this. The transition probabilities shown in Figure 4 and reported in Table 2 are obtained as combinations of the basic events probabilities reported in Table 1. The Markov chain in Figure 4, is organised into five levels plus the final states (final states are identified with H, SF and CF). Level one involves the hardware aspects, the level two the diagnostic tests carried out on boards and communications channels (hardware). Level three checks databases; level four investigates the behaviour of the acceptance test on the keyboard input whereas level five involves software. It should be noted that the model doesn't represent the timing relations among the various events.

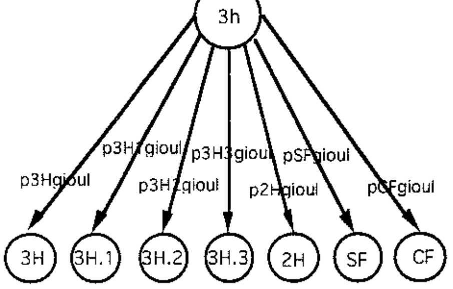

Figure 5: Compact description of the behaviour of GIOUL in configuration 3h

Probability notation	Meaning
p3Hgioul	success with three correct GIOUL units
p3Hxgioul	success with three GIOUL units, but the unit x disagreed
p2Hgioul	success with two GIOUL units
pFBgioul	safe GIOUL unit failure
pFCgioul	catastrophic GIOUL unit failure

Table 4: Notation of the transition probabilities of Figure 5

The upper level (level 2 of Figure 3) is not concerned with this detailed view, from its perspective it is just necessary to observe that the GIOUL TMR, performing one execution, can jump from the initial 3h state into all the other GIOUL TMR configurations (success state) or into the failure (safe or catastrophic) state according to the transition probabilities resulting by the resolution of model in Figure 4. Thus its view can be represented by the Markov chain in Figure 5 where all the paths, that go from the initial state to each of the final states, are substituted with one arch and the relative transition probability (reported in Table 4).

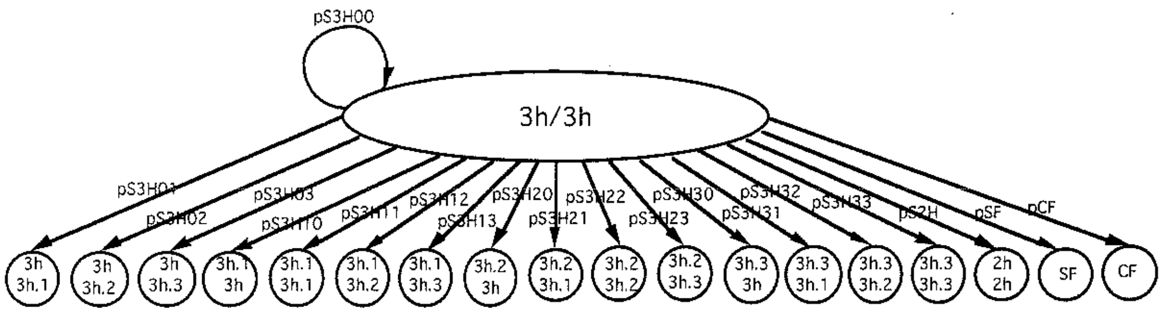

Figure 6: Black-box model of one execution from the configuration 3h/3h

Probability notation	Meaning
pS3H00	success with three correct sections
pS3H0y	success with three yet active sections, but the GP y unit disagreed
pS3Hx0	success with three yet active sections, but the GIOUL x unit disagreed
pS3Hxy	success with three yet active sections, but the GP y unit and GIOUL x unit disagreed
pS2H	success, but one section is excluded
pSF	the system is stopped
pCF	catastrophic failure

Table 5: Notation of the transition probabilities of Figure 6

Once all the models of level 1 have been obtained and the related transition probabilities computed, these objects are composed into the several models for one system execution (level 2 Figure 3). Also for this level, only the description of one of the 17 configurations is provided; the remaining 16 can easily be obtained by analogy. This time we proceed in a top-down approach to show our models by showing first what is the viewpoint of the next level, the mission level (level 3 of Figure 3). Figure 6 depicts the black-box model of configuration 3h/3h showing just the transitions to other configurations after one execution and their probabilities (reported in Table 5).

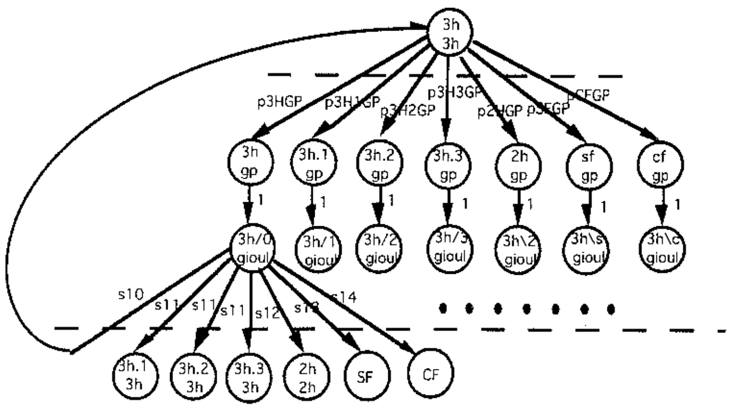

Figure 7: Partial explosion of the model of Figure 6

State notation	Meaning
3h 3h	GIOUL and GP success with the GIOUL and GP correct units; start execution of four GP loops
3h gp	the four GP loops terminated with three GP correct units
3h.x gp	the four GP loops terminated, but the GP units of the section x disagreed
2h gp	the four GP loops terminated with only two GP units already active
fb gp	safe GP unit failure
fc gp	catastrophic GP unit failure
3h/0 gioul	GP not disagreeing and start checking the GIOUL outputs
3h/1 gioul	the GP unit of section x disagreed and start checking the GIOUL outputs
3h\2 gioul	a GP unit is already excluded and start checking the GIOUL outputs
3h\s gioul	safe GP unit failure and and start checking the GIOUL outputs
3h\c gioul	catastrophic GP unit failure and start checking the GIOUL outputs
3h.x 3h	GIOUL and GP success, but the GIOUL unit of the section x disagreed
2h	GIOUL and GP success, but a GIOUL unit is excluded
SF	the GP units are correct, but the GIOUL units failed safely causing the safe system failure
CF	the GP units are correct, but the GIOUL units failed catastrophically causing the catastrophic system failure

Table 6: Notation of the states of Figure 7

Probability notation	Meaning
p3HGP	after four loops, the GP units are correct
p3HxGP	after four loops, the GP units are working, but the GP unit of the section x already disagreed
p2HGP	after four loops, GP succeeds, but a unit is excluded
pSFGP	safe GP unit failure
pCFGP	catastrophic GP unit failure
p3Hgioul	s10
p3H1gioul	s11
p2Hgioul	s12
pSFgioul	s13
pCFgioul	s14

Table 7: Notation of the transition probabilities of Figure 7

Of course, in order to obtain the transition probabilities mentioned above it is necessary to explode the arcs which go from the "3h/3h" state to the final states and describe the system behaviour when one iteration starts with the GIOUL and GP TMR working perfectly. This explosion is shown in Figure 7 (between the dashed lines); it should be noted that the graph is not complete in order to leave the figure clear, still it is easy to deduce the entire model. The notation of the transition probabilities of Figure 7 is reported in Table 7 and the notation of the states in reported in Table 6.

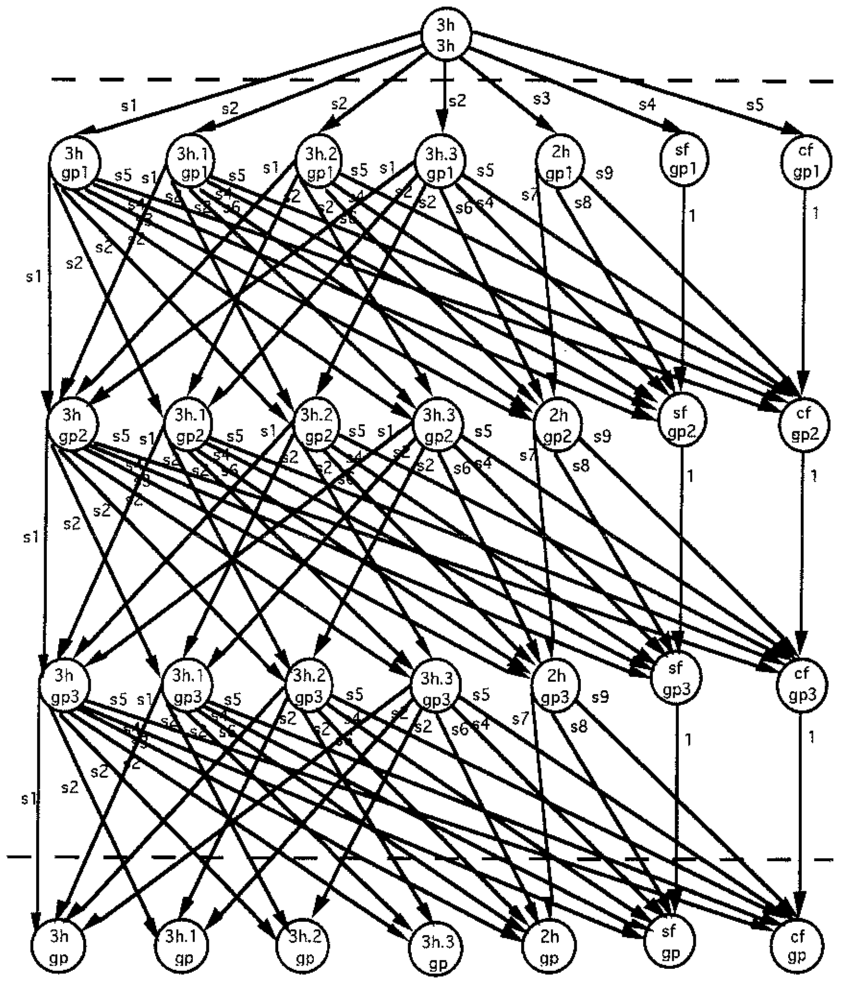

Figure 8: Model for the four executions of GP in each system execution

Probability notation	Meaning
s1 = p3Hgp	the three GP units are correct during the short loop
s2 = p3H1gp	the GP unit of section x disagreed once
s3 = p2Hgp	one section is excluded with any unit not disagreeing at the loop start
s4 = pFBgp	safe GP one-execution failure
s5 =pFCgp	catastrophic GP one-execution failure
s6 = p2Hgpi	one section is excluded due to two successive GP unit disagreeing
s7 =p22Hgp	success with only two active GP units
s8 =p2FBgp	safe failure with only two active GP units
s9 = p2FCgp	catastrophic failure with only two active GP units

Table 8: Notation of the transition probabilities of Figure 8

State notation	Meaning
3h gp0	GP execution: first short loop
3h gp1	during the first short loop, the GP units are correct; GP execution: second short loop
3h.x gp1	during the first short loop, the GP unit of the section x disagreed; GP execution: second short loop
2h gp1	during the first short loop, a GP unit is excluded; GP execution: second short loop
fb gp1	during the first short loop, safe GP unit failure
fc gp1	during the first short loop, catastrophic GP unit failure
3h gp2	during the second short loop, the GP unit of the section x disagreed; GP execution: third short loop
3h.x gp2	during the second short loop, the GP unit of the section x disagreed; GP execution: third short loop
2h gp2	during the first or second short loop, a GP unit is excluded; GP execution: third short loop
fb gp2	during the first or second short loop, a GP unit is excluded; GP execution: third short loop
fc gp2	during the first or second short loop, catastrophic GP unit failure
3h gp3	during the third short loops, the GP unit are correct;GP execution: fourth short loop
3h.x gp3	during the third short loop, the GP unit of the section x disagreed; GP execution: fourth short loop
2h gp3	during the first or second or third short loop, a GP unit is excluded; GP execution: fourth short loop
fb gp3	during the first or second or third short loop, safe GP unit failure
fc gp3	during the first or second or third short loop, catastrophic GP unit failure

Table 9: Notation of the states of Figure 8

The transition probabilities (pxxGP) from the state "3h/3h" represent the output transition probabilities for the four executions of GP which may result in the different outcomes represented by the different states following by one execution of GIOUL. The transition probabilities from the state "3h/3h" are obtained from the model (sub-level of level 2) shown in Figure 8; note that each state (notations in Table 9), together with the corresponding transition probability (notations in Table 9), represents, in Figure 8, one of the object already studied in level 1.

The system model obtained with Ultrasan allows to represent all the 17 configurations of one execution. Also in modelling using Ultrasan we started following a modular approach, building first the basic objects (level 1) and then putting them together in the system model for one execution. Unfortunately, however, the valuable possibility offered by Ultrasan to define separated models and to join them into the Ultrasan "Composed Model" is very useful for conceiving a design but slows down the execution speed of the compound model. In fact the common places between sub-models must be attached to timed transitions, and this is useless for this model. This causes the increment of the state number in the "Reduced Model" and decreases the evaluation speed. Thus we decided not to take advantage of this opportunity but preferred to speed up the evaluations as much as possible. Figure 9 shows the model that we actually used. Despite it can just provide an idea of the size and complexity, the existence of two sub-models is visible: the upper part represents one iteration of GP and it is executed four times, the lower part represents one iteration of GIOUL executed once. An additional general problem to the understanding of the behaviour of models built using Ultrasan derives from the extensive use of C code that is hidden into the gates.(This model was evaluated using the Ultrasan transient solver)

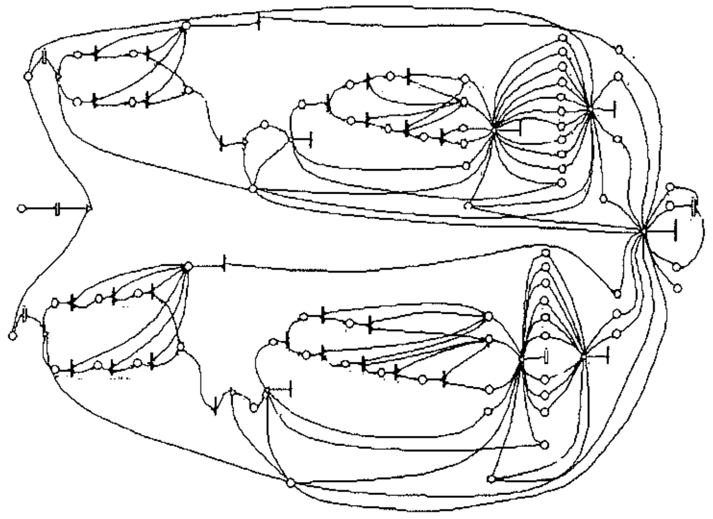

Figure 9: Ultrasan model for one execution

5. Models for the mission and wanted measures.

Also the model for a mission has been developed using both Markov chains and SANs. Despite the constant time of one second required for each system execution, modelling a mission using discrete time Markov chains (MC/TD) is not feasible due the following reasons:

1) The system can operate with only two active sections (2h/2h), if no failures occur, for a time interval in the range 5÷45 minutes (15 minutes average) i.e. a much longer time and a very large number of states "2h/2h";

2) Since the model has to be as general as possible, the time to restore a section should be left as a parameter and this cannot be accomplished using a discrete time model.

For all these reasons continuos time Markov Chains (MC/TC) have been used. The obtained model solves the above problems, but it is approximate. In fact not only it uses exponential distributions in place of the deterministic ones (this is not a problem since we use a very long mission time with respect to the time required from one execution), but the main approximation comes from the fact that the n states "2h/2h" in the hypothetical MC/TD are compressed in only one state "2h/2h". The output rates of this compressed "2h/2h" state approximate the behaviour (averaging both time and probabilities)[16]. The MC/TC model defined is partially depicted in Figure 10 only to give an idea on the state transitions. This model allows to evaluate the probabilities of safe or catastrophic failure of missions of a given duration. The model has been solved obtaining the system of differential equations [21]. The solution was obtained using Mathematica: it provided both symbolic and numerical results. The solutions we have found are complete since they allow to evaluate the probability of safe or catastrophic failure over any given time interval t.

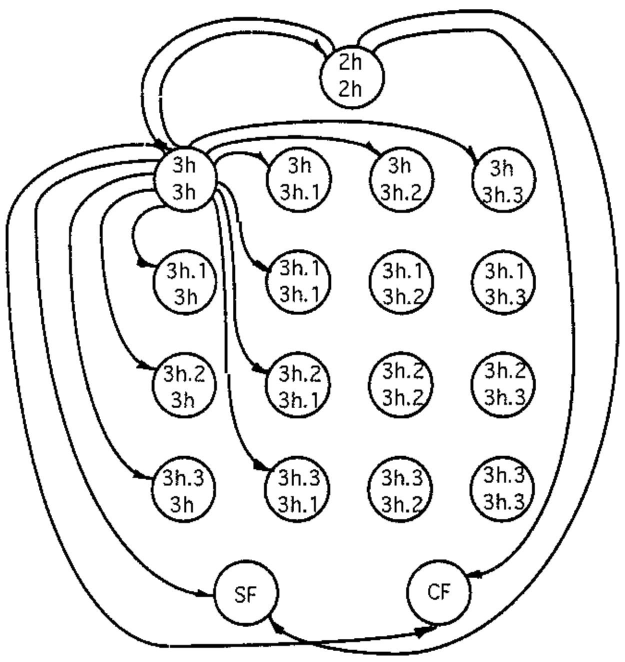

Figure 10: Partial representation of the MC/TC mission model

To validate this approximated model we compared the results obtained with those returned by a mission model built using Ultrasan (where time distribution is still exponential). The Ultrasan model is structured into two layers: the lower layer is composed by a number of sub-models each one representing a given system configuration, and an higher level (“Composed Model”) that joins the sub-models as required. For the model of a mission we did use the Composed Model without slowing down the execution speed because each sub-model already contained timed transitions at the beginning. In Figure 11a the sub-model representing the configuration "3h/3h" is shown: the other sub-models are very similar. In Figure 11b the "Composed Model" is shown; each box indicates one sub-model of one-execution and the "join" box links them together.

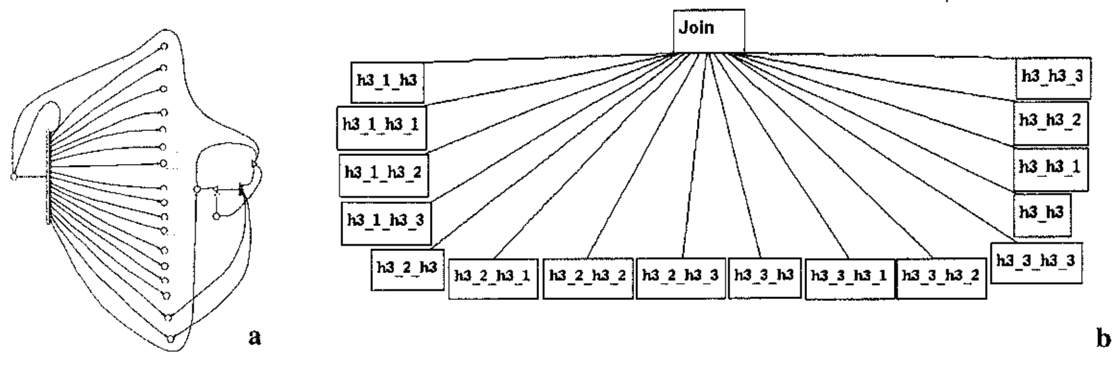

Figure 11: Ultrasan a) 3h/3h sub-model b) "Composed" mission model

The two mission models (with the Markov chains and with SANs) have been tested against each other on the same input data for variable mission duration up to one year. The results provided are of the same magnitude as soon as t reaches 1000 seconds, and for t from one day up they can be considered identical; as t grows, the number of identical significant digits increases and even exceeds the desired accuracy. While we preferred to use the Ultrasan model to obtain

the results for one execution (and used a Markov one to validate it) because of the possibility to represent all the different configurations within a single model, here we have done the opposite: we used the model based on Markov to compute the results and the one based on SANs to validate it. We preferred to use Mathematica and the Markov model to achieve results at a higher speed: only few minutes are required to provide the results for one year missions while our Ultrasan model requires several days.

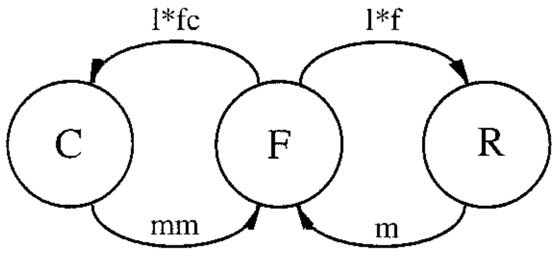

Figure 12: The availability model

Once the probabilities of success or failure (safe and catastrophic) in one year have been obtained, it is immediate to find out the reliability and safety measures while to obtain availability measures it is necessary build an availability model which is represented by the continuos Markov Chain (MC/TC) illustrated in Figure 12. When the system is in state F, it is operating and provides a correct service. R and C indicate the repair states: the system is not available following a safe failure (R) or a catastrophic failure (C). The system resuming rates are "m", following a safe failure, and "mm", following a catastrophic failure. "l" indicates the failure transition rate; it should be multiplied for "f", the probability of safe failure, or for "fc", the probability of catastrophic failure. This availability model (with repair after catastrophic failure) respects the behaviour of the TMR MODIAC; in fact, in a railway interlocking system, a catastrophic event involves usually only one part of the controlled system, while the rest of the system must continue working also after a catastrophic event.

6. Evaluations

Once the models have been realised, the system behaviour can be evaluated, to check for example if the system meets its requirements, and to analyse the sensitivity to the various parameters. The input variables for the set of models can be split in those necessary to the models representing one execution and those for models of the mission. The models of one execution require the probability values of the basic events (described in Table 1) while those related to the mission require i) the results of the evaluation of one execution, ii) the repair time for restoring an excluded section, iii) the mission time, and iv) the recovery time (only for the availability model). Since there are many input variables, reasonable values have been "a priori" assigned to all the parameters, and the sensitivity of the dependability measures has been investigated for just one parameter at a time: sensitivity first to the main parameters (hardware, software), then to the secondary parameters have been evaluated; finally, a specific study on software has been carried out. Due to space limitations we do not report on all the evaluations carried out [16] but show just a few examples.

The probability values of the basic events are reported in Table 11, while we set the mission duration to one year, the failure rate l which has to be multiplied by the failure probability to one

per hour, the repair rate following a safe failure, m, to 2 per hour and the repair rate following a catastrophic failure, mm, to 1/2 per hour. Moreover the correlated error between two units (q2c) has been expressed as a function of the independent error (qiv): q2c = corr qiv^2, where "corr" has been set to 100: this corresponds to the assumption of a positive correlation among software errors.

Probabilities	Values/hour	Probabilities	Values/hour
qhl	8E-5	qhp	2E-5
qhdl	2E-10	qhdp	2E-10
qhndl	3E-12	qhndp	3E-12
qbrl	1E-8	qbrp	1E-8
qbnrl	1E-10	qbnrp	1E-10
qil	1E-5	qip	1E-5
qail	1E-8	.....	.....

Table 10 : GIOUL and GP parameters

6.1 Measurements with variable software error probability .

Figures 13, 14 and 15 show dependability measurements as a function of the probability of software independent error. The range goes from 1E-3 to 1E-6 per hour; curves for different values of the hardware error probability per unit are reported.

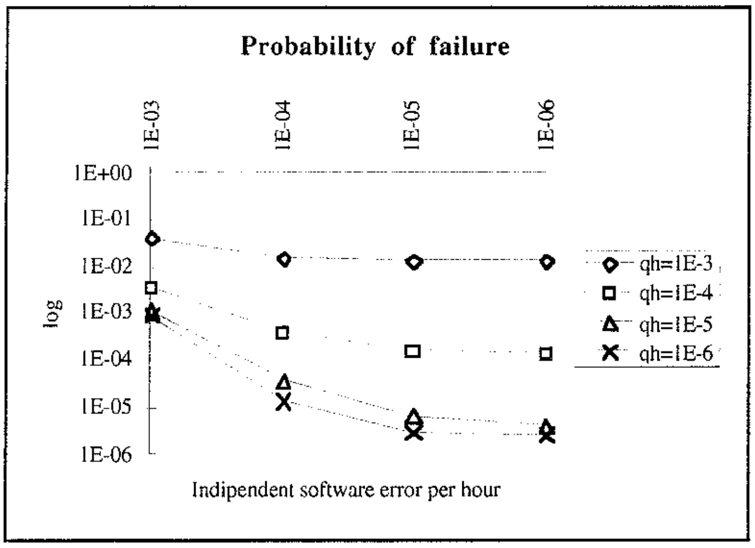

Figure 13: Failure probability (one year mission)

Observing the shapes of the curves in Figure 13, it is clear that reliability is sensitive to variations of the software error probability if hardware quality is good enough (ranging from 1E-5 to 1E-6), instead it becomes more and more insensitive as the hardware error probability increases. The curves also point out that decreasing the probability of software error over 1E-5 (that is improving the quality of the software) is practically useless without decreasing all the remaining system parameters at the same time as well. In fact, while in the left side of the fig-

ure, from 1E-3 to 1E-5, it can be observed that the reliability improves, it remains approximately constant for the values in the right side.

The shape of the curves representing the probability of catastrophic failure, shown in Figure 14, appears quite different. Safety is almost a "linear" function of the probability of independent software error: the more the software error probability decreases and the more the safety improves. Of course, this is valid under certain conditions, that is as long as the remaining parameters are kept unchanged and the software error probability is kept over a certain value. The same Figure also points out that the safety seems completely insensitive to variations of the hardware error probability: in fact, it is difficult to distinguish among the four curves shown. It should also be noted that the system satisfies its target (that for such systems is usually 1E-7 per year) if independent error probability values are less than or equal to 1E-5.

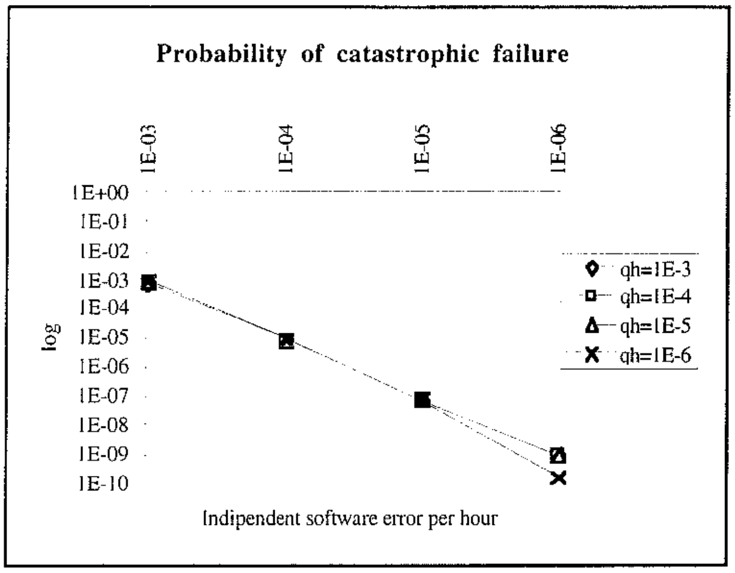

Figure 14: Catastrophic failure probability (one year mission)

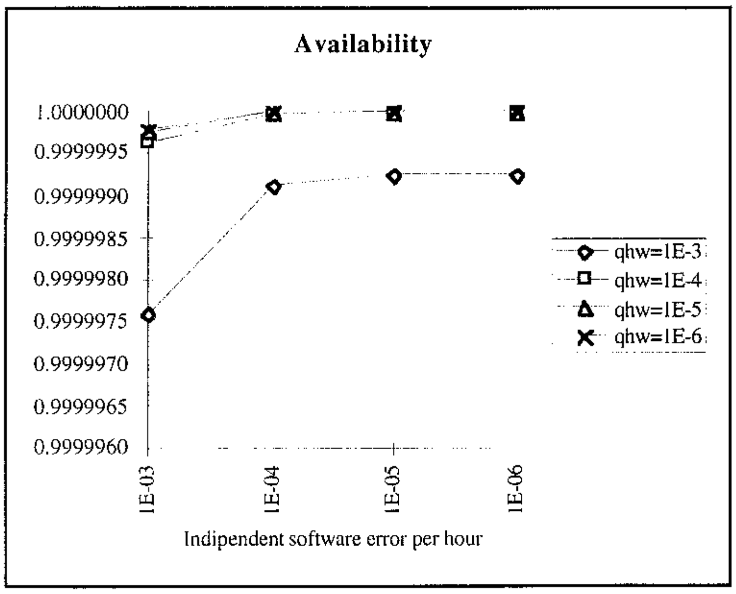

Figure 15: Availability (one year mission)

Figure 15 shows the availability, it clearly points out that the availability is very high and almost constant when the software error probability ranges form 1E-4 to 1E-6, while it is a bit worse for higher values. The figure shows also that there are almost no variations of the availability for values of the hardware error probability equal or better than 1E-4. In any case the system is never affected by availability lacks, since the target (5 minutes unavailability over 8600 hours, i.e. 0.999990310078) is satisfied in all the considered range of the software error probability. In short, within our parameters setting, the software must be regarded as a critical factor for safety, while it appears of almost no concern for availability.

6.2 *Measurements with variable hardware error probability.*

Figures 16, 17 and 18 show dependability measurements versus independent hardware error probability per unit ranging from 1E-3 to 1E-6; curves are quoted as a function of the independent software error probability per section.

Figure 16 shows the failure probability decreasing in a linear fashion as the hardware error probability decreases too, its slope also decreases as it approaches the value 1E-5. This points out that, increasing hardware reliability, the overall system reliability also slightly increases (if hardware error probability is reduced by an order of magnitude, then the overall system reliability is increased of two orders of magnitude), but it is not necessary to exceed certain values without improving software and/or database quality as well. Figure 16 also points out that, when software error probability is good enough, the curves are quite closer: this implies that software is not a critical factor at all for system reliability if the error probability ranges into reasonable values.

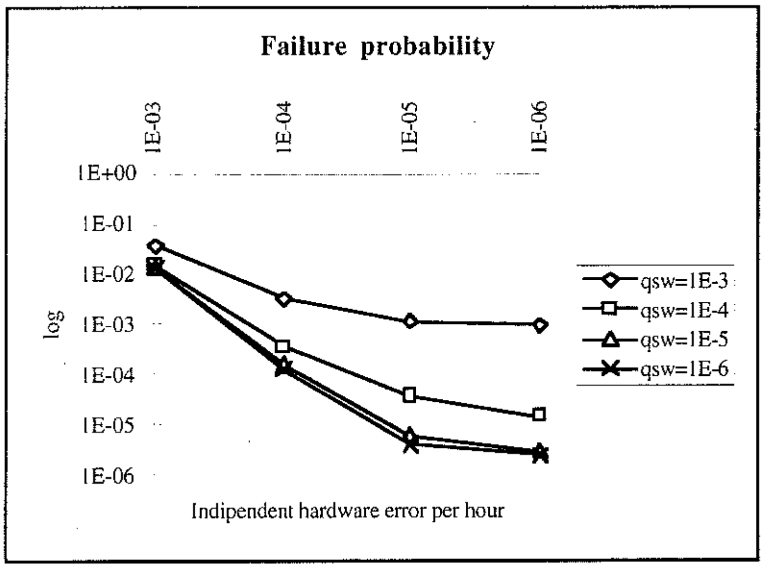

Figure 16: Failure probability (one year mission)

Figure 17 shows that safety is quite insensitive to hardware error probability variations, instead it is strongly sensitive to software error probability variations.

Figure 18 shows availability versus hardware error probability. Observing the curve shapes within a certain hardware error probability range, no availability lacks are remarkable; neverthe-

less any hardware error probability increase can result harmful, as we deduce from the curve slope regarding values higher then 1E-4.

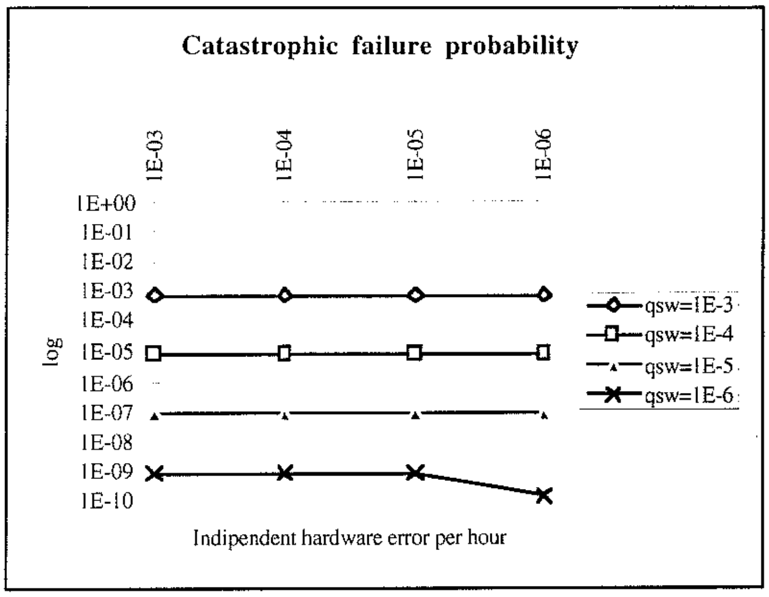

Figure 17: Catastrophic failure probability (one year mission)

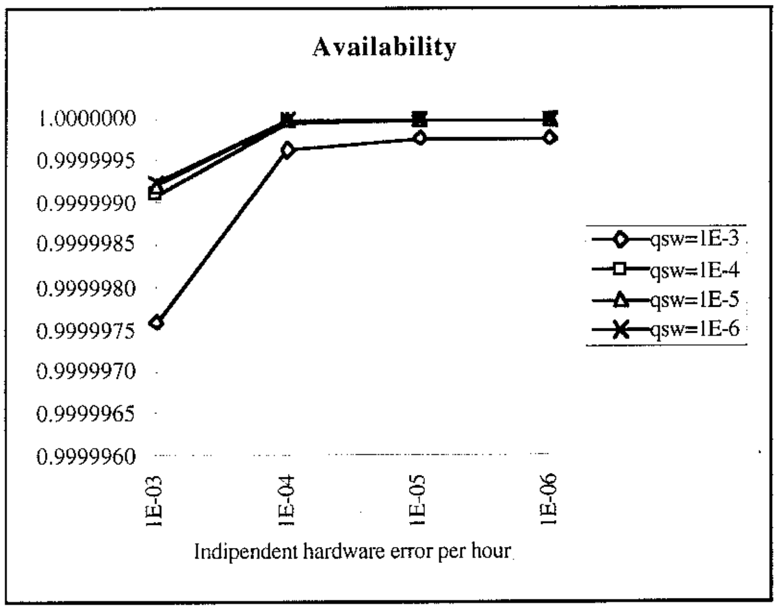

Figure 18: Availability (one year mission)

6.3 Measurements with variable database error probability.

It should be recalled that database error probability consists of detected error probability plus not-detected error probability. In order to achieve the dependability measurements, the above two probabilities have been linked: the former has been assumed about two orders of magnitude greater then the latter. This is indeed an arbitrary choice, but yet appears realistic. Measurements have been made with regard to two independent software error probability values (1E-3 and 1E-5), whereas database error probability is in the range 1E-4 to 1E-10.

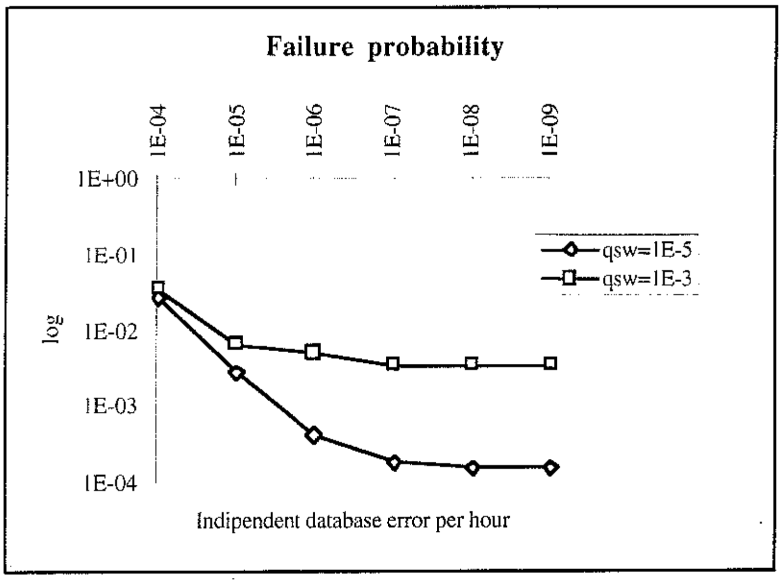

Figure 19: Failure probability (one year mission)

Figure 19 clearly shows how a software error probability increase makes the failure probability grow by two orders of magnitude when database error probability is less then 1E-6, whereas such difference decreases, when this gets larger. This implies that, no matter how good the database is (always within an acceptable range), reliability cannot improve if software is not reliable. Another aspect to be pointed out in Figure 19 is the different slope of the two curves: when qsw equals 1E-3 the curve appears less sensitive to database quality improvements and almost instantaneously reaches a steady state; instead, when qsw equals 1E-5, the slope is greater and more gradual.

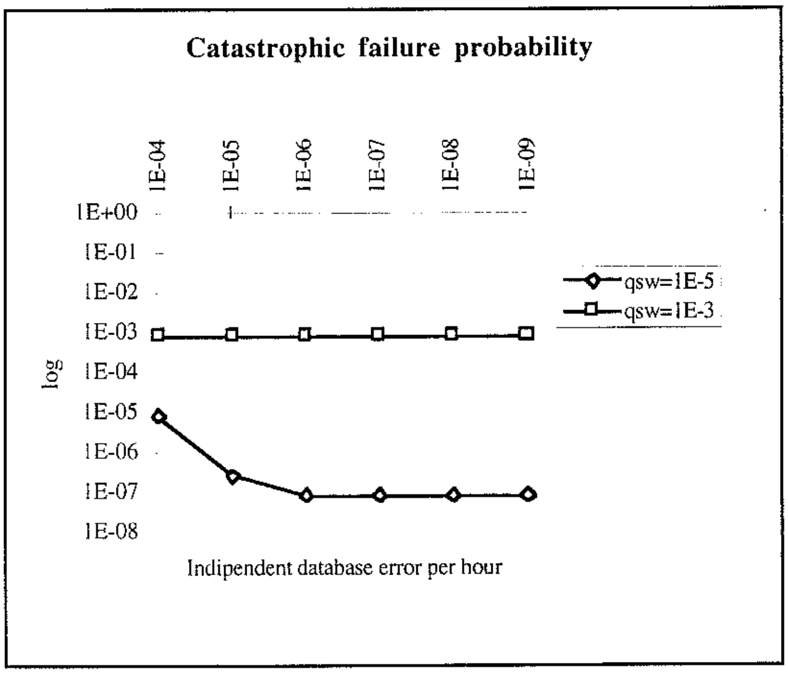

Figure 20: Catastrophic failure probability (one year mission)

Figure 20 shows that safety is almost insensitive to database error probability variations, its

corresponding value being unchanged, when qsw equals to 1E-3: the system could not satisfy the target, even though it made use of totally reliable databases. Instead, it is clear that, if software is good, database can result a discriminating factor for safety. In fact, if database quality is poor, then catastrophic failure probability can grow up.

Figure 21 shows availability versus database error probability. The two curve shapes appear quite similar, even if availability increases as a result of a qsw decrease. Targets are satisfied, but the curves indicate that unavailability can sensibly increase if one of the two parameters grows worse.

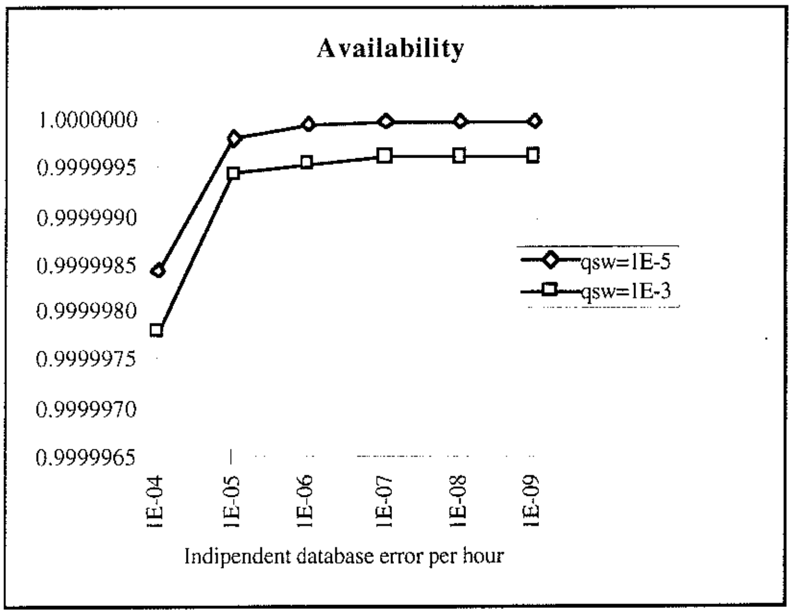

Figure 21: Availability (one year mission)

6.4 Specific study on software: correlation coefficient variation.

This section shows how dependability measurements are affected when changing the relationship linking the correlated error probability to the independent error probability (per execution). The coefficient (corr) has been made varying in the range 1E0 to 1E4, whereas the software error probability has assumed the values 1E-3, 1E-4 and 1E-5.

Figure 22 clearly shows the different shapes assumed by reliability depending on software error probability and "corr" factor: in fact, if qsw equals 1E-5 then "corr" factor does not have any effects on failure probability, instead, as the software error probability grows, an possible correlation factor increase can drop the reliability. It is also evident the influence on system reliability of higher software error probability is high and higher growth of the correlation factor.

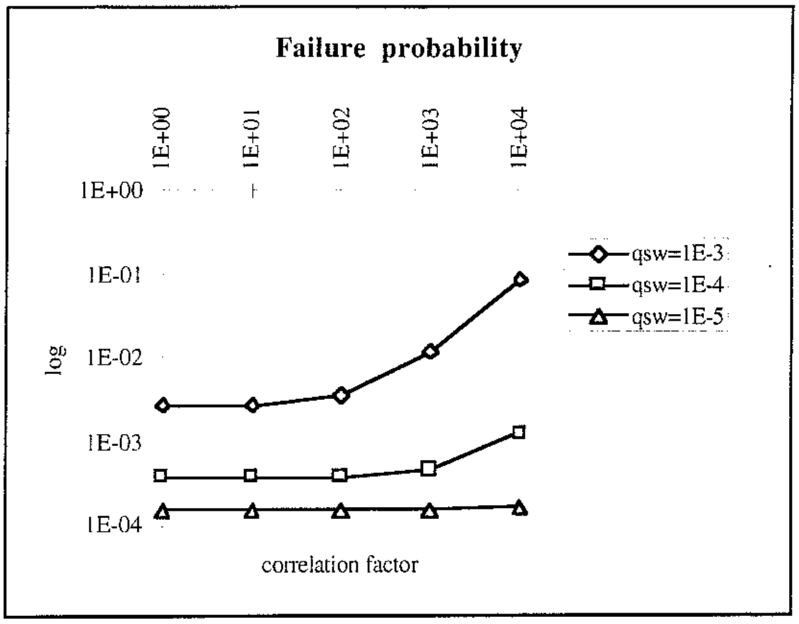

Figure 22: Failure probability (one year mission)

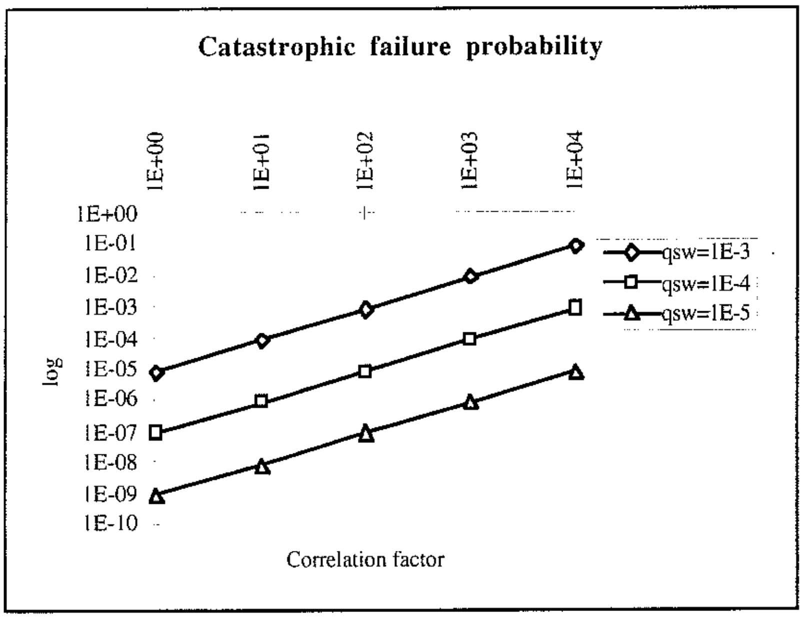

Figure 23: Catastrophic failure probability (one year mission)

Again, with regards to the three qsw values mentioned above, safety always varies proportionally to the correlation coefficient. As a result, the three curves appear parallel, as shown in Figure 23. One important aspect should be pointed out: the safety target is 1E-11 referred to a hour, which means 1E-7 if referred to a year; thus, all the possible qsw and "corr" combinations that make the safety target satisfied can be obtained drawing the curves intersections across a horizontal line corresponding to a safety value of 1E-7. This target is actually satisfied by qsw equal to 1E-5 and corr less then or equal to 100, and qsw equal to 1E-4 and corr equal to 1 as well: this implies that a software with an error probability reduced by an order of magnitude can be used still avoiding serious consequences, only if the correlation factor is small enough. Of course, if qsw increases too much (1E-3), then the safety is sensibly

modelling method and evaluation tool for each level. Last, and very important, in order to validate the most complex models, we built alternatives using different tools and methodologies; this constitutes an example of application of diversity and proved to be very useful since it allowed to find small bugs and imperfections and to gain more confidence that the models represented the real system behaviour.

With respect to the evaluated characteristics of the Ansaldo TMR MODIAC, our analyses, which have not been reported here, allowed to establish the dependability bottlenecks of the current system and to state targets for the several sub-components such that the system targets could be reached. We could thus provide hints for next releases or modifications of the systems and information to assign targets, and consequently the budget, to the various components of the system. The work presented in this paper constitutes a nucleus that will be expanded in further studies: some directions are, the release of some of the simplifying hypotheses or the modelling of some variation of the system characteristics to better identify directions for next releases. Another step to be also carried on is the modelling of the rest of the system, to evaluate the dependability figures accounting for the trackside units and the communication network.

References

[1] J. Arlat, K. Kanoun and J. C. Laprie, "Dependability Modelling and Evaluation of Software Fault-Tolerant Systems," IEEE Transaction on Computer, Vol. 39, pp. 504-513, 1990.

[2] A. Avizienis and J. P. J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments," IEEE Computer, Vol. pp. 1984.

[3] J.B. Balliet and J.R. Hoelscher, "Microprocessor based Interlocking Control - Concept to Application," in Proc. APTA Rail Transit Conf., Miami, Fl., 1986, pp. 13.

[4] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico and S. La Torre, "Dependability of Iterative Software: a Model for Evaluating the Effects of Input Correlation," in Proc. accepted at SAFECOMP '95, Belgirate, Italy, 1995, pp.

[5] S. Chiaradonna, A. Bondavalli and L. Strigini, "On Performability Modeling and Evaluation of Software Fault Tolerance Structures," in Proc. EDCC1, Berlin, Germany, 1994, pp. 97-114.

[6] A. Costes, C. Landrault and J. C. Laprie, "Reliability and Availability Models for Maintained Systems featuring Hardware Failures and Design Faults," IEEE Trans. on Computers, Vol. C-27, pp. 548-60, 1978.

[7] A.H. Cribbens, M.J. Furniss and H.A. Ryland, "The Solid State Interlocking Project," in Proc. IRSE Symposium "Railway in the Electronic Age", London, UK, 1981, pp. 1-5.

[8] F. Di Giandomenico, A. Bondavalli and J. Xu, "Hardware and Software Fault Tolerance: Adaptive Architectures in Distributed Computing Environments," Esprit BRA 6362 PDCS2 Technical Report, june 26 1995.

[9] J. B. Dugan and M. Lyu, "System-level Reliability and Sensivity Analysis for Three Fault-Tolerant Architectures," in Proc. 4th IFIP Int. Conference on Dependable Computing for Critical Applications, San Diego, 1994, pp. 295-307.

[10] E. K. Holt, "The Application of Microprocessors to Interlocking Logic," in Proc. APTA Rail Transit Conf., Miami, Fl., 1986, pp. 13.

[11] K. Kanoun, M. Borrel, T. Morteveille and A. Peytavin, "Modelling the Dependability of CAUTRA, a Subset of the French Air Traffic Control System," LAAS Report, December 1995.

[12] J. C. Laprie, C. Beounes, M. Kaaniche and K. Kanoun, "The Transformation Approach

to Modeling and Evaluation of Reliability and Availability Growth of Systems," in Proc. 20th IEEE Int. Symposium on fault Tolerant Computing, Newcastle, UK, 1990, pp. 364-71.

[13] J.C. Laprie and K. Kanoun, "X-ware Reliability and Availability modelling," IEEE Trans. on Software Engineering, Vol. SE-18, pp. 130-147, 1992.

[14] G. Mongardi, "A.C.C Specifiche Tecniche e Funzionali," Ansaldo Trasporti

[15] G. Mongardi, "Dependable Computing for Railway Control Systems," in Proc. DCCA-3, Mondello, Italy, 1993, pp. 255-277.

[16] M. Nelli, "Modellamento e valutazione di attributi della dependability di un sistema critico per l'interlocking ferroviario", Tesi di Laurea, Facolta' di Ingegneria, University of Pisa, Pisa, 1995.

[17] D. Nordenfors and A. Sjoeberg, "Computer Controlled Electronic Interlocking System, ERILOCK 850," ERICSSON Review, Vol. 1, pp. 1-12, 1986.

[18] I. Okumura, "Electronic Interlocking to be tried in Japan," Railway Gazette International, Vol. 12, pp. 1043-1046, 1980.

[19] H. Strelow and H. Uebel, "Das Sichere Mikrocomputersystem SIMIS," Signal und Draht, Vol. 4, pp. 82-86, 1978.

[20] A. T. Tai, A. Avizienis and J. F. Meyer, "Evaluation of fault tolerant software: a performability modeling approach," in "Dependable Computing for Critical Applications 3", C. E. Landwher, B. Randell and L. Simoncini Ed., Springer-Verlag, 1992, pp. 113-135.

[21] K. S. Trivedi, "Probability and Statistics with Reliability, Queuing, and Computer Science Applications," Durham, North Carolina, Prentice-Hall, Inc., Englewood Cliffs, 1982.

[22] G. Wirthumer, "VOTRICS - Fault Tolerant realised in software," in Proc. SAFECOMP, Vienna, Austria, 1989, pp. 135-140.